# Texture Analysis with Arbitrarily Oriented Morphological Opening and Closing

Jan Bartovsky, Eva Dokladalova, Michel Bilodeau, Petr Dokládal

# Texture Analysis with Arbitrarily Oriented Morphological Opening and Closing

Jan Bartovský, Eva Dokládalová, Michel Bilodeau and Petr Dokládal

Internal Report N-14/11/MM

*available at:*
http://hal-ensmp.archives-ouvertes.fr/ENSMP/hal-00651251

Centre of Mathematical Morphology, Mines-ParisTech,
35, rue Saint Honoré, 77300 Fontainebleau Cedex

December 15, 2011

**Abstract**

This paper presents a fast, streaming algorithm for 1-D morphological opening on 2-D support. The algorithm is further extended to compute the complete size distribution during a single image run. The Structuring Element (SE) can be oriented under arbitrary angle that allows us to perform different orientation-involved image analysis, such as local angle extraction, directional granulometries, etc.

The algorithm processes an image in constant time irrespective of the SE orientation and size, with a minimal latency and very low memory requirements. Regardless the SE orientation, it reads and writes data strictly sequentially in the horizontal scan order. Aforementioned properties allow an efficient implementation in embedded hardware platforms that opens a new opportunity of a parallel computation, and consequently, a significant speed-up.

**Keywords:** Mathematical Morphology, Granulometry, parallel implementation

# 1 Introduction

During several last decades, the problems of object recognition, texture analysis, orientation, and size distribution have become very important in image analysis and computer vision field. Tthe Mathematical Morphology (MM) has proved to be a very powerful tool in solving such problems [1] [2]. On the other hand, the computation complexity of profound analysis over many shapes and rotations is overwhelming as MM often resides on iterative use of two basic operations dilation and erosion.

The paper is organized as follows: Section 2 recalls the basic notion of morphological operators. Section 3 outlines the main principle of the proposed algorithm. Section 4 presents performance results of computer benchmarks and a comparison with other algorithms. Finally, Section 5 displays a few examples of opening and size spectrum applications.

## 1.1 State of the Art

One of the early algorithms of morphological erosion that complete in $\mathcal{O}(1)$ per pixel is van Herk [3]. Although it runs in constant time using only 3 comparisons per pixel, it requires two passes: direct and reverse, which degrades its real-time features especially if extended to 2-D. Later, Gil and Kimmel [4] improved van Herk algorithm and reduced the number of comparison to only 1.5 per pixel. Soille *et al.* [5] first published the principle of arbitrary oriented SE. The SE rotation is handled by Bresenham algorithm, the computation is carried out by the van Herk algorithm.

Van Droogenbroeck and Buckley [6] proposed a different approach based on notion of anchors, the points that are not affected by a transformation. This algorithm uses a histogram for calculation making any implementation dependent on the number of gray levels.

Urbach and Wilkinson [7] proposed an algorithm for arbitrary shaped 2-D flat SE based on the computation of multiple horizontal line SEs for every pixel and storing them in a lookup table. The result is then computed by taking the maximum from the lookup table corresponding to the shape of the SE.

The issue of granulometry has been already targeted by Vincent [8] and Menotti-Gomes *et al.* [9]. Both computed the size distribution and opening trees in linear time with respect to the size of the image.

## 1.2 Novelty

We propose a new algorithm that computes both the opening and size spectrum at the same time with the following properties. The algorithm supports line SE oriented in arbitrary angle. It processes an image in constant time with respect to length of SE or its orientation. It accesses to the input and output data strictly sequentially in a horizontal scan order inferring a small computation latency. This derandomization of memory bandwidth along with very low memory requirements, which are much lower than the mere size of the image to process, make the algorithm especially interesting for real-time hardware systems. The single-thread C implementation of this algorithm brings a very low execution time and outperforms a few other existing efficient algorithms in the case of opening. In the case of size spectrum, the speed-up against the conventional approach, which requires many openings to be computed and then subtracted (4), is in orders of magnitude.

## 2 Basic Notions

Let $\delta_B$, $\varepsilon_B$: $\mathbb{Z}^2 \to \mathbb{R}$ be a dilation and an erosion on gray-scale images, parameterized by a SE $B$ assumed flat (i.e., $B \subset \mathbb{Z}^2$) and translation-invariant, defined as

$$\delta_B(f) = \bigvee_{b \in B} f_b \; ; \qquad \varepsilon_B(f) = \bigwedge_{b \in \widehat{B}} f_b \; . \tag{1}$$

The hat $\widehat{\ }$ denotes the transposition of the structuring element, equal to the set reflection $\widehat{B} = \{x \mid -x \in B\}$, and $f_b$ denotes the translation of the function $f$ by some scalar $b$. The SE $B$ is equipped with an origin $x \in B$.

Let $\varphi_B$, $\gamma_B$: $\mathbb{Z}^2 \to \mathbb{R}$ be a closing and an opening on gray-scale images, parameterized by the aforementioned SE $B$, defined as

$$\varphi_B(f) = \varepsilon_B[\delta_B(f)] \; ; \qquad \gamma_B(f) = \delta_B[\varepsilon_B(f)] \; . \tag{2}$$

Let $S_{\lambda B}$: $\mathbb{R}^2 \to \mathbb{R}$ be a size spectrum, parametrized by a SE $B \subset \mathbb{R}^2$ and its size $\lambda$, defined as (originally proposed by Maragos [2])

$$S_{\lambda B}(f) = -\frac{d}{d\lambda} \| \gamma_{\lambda B} f \|; \quad f : \mathbb{R}^2 \to \mathbb{R} \tag{3}$$

Since we are interested in images with a discrete and bounded support $D \subset \mathbb{Z}^2$, $D = [1..M] \times [1..N]$, the discrete size spectrum $S_{\lambda B}$ is transformed to as follows

$$S_{\lambda B}(f) = \sum_D \left( \gamma_{(\lambda-1)B} f - \gamma_{\lambda B} f \right); \quad f : D \to \mathbb{R}, \tag{4}$$

considering the size spectrum step $d\lambda = 1$.

Hereafter, we focus our description on the opening by a line SE $B^\alpha_\lambda$. This SE has shape of a discrete line of length $\lambda$ rotated by angle $\alpha$ from positive x-axis counter-clockwise. The closing can be obtained in accordance to the duality property.

The oriented 1-D opening can also be used for detection of local orientation (orientation field) $\zeta_\lambda(f)$, see (5), image restoration, or oriented spectrum (6). The oriented size spectrum $OS(\alpha, \lambda)$ of an anisotropic texture is the size distribution expectancy of a 1-D signal obtained by intersection with a randomly drawn straight line. The expectancy is approximated by the frequency count in (4).

$$\zeta_\lambda(f) = \arg \max_{\alpha \in (0,180)} \gamma^\alpha_\lambda(f) \tag{5}$$

$$[OS(\alpha, \lambda)](f) = \sum_D \left( \gamma_{B^\alpha_{\lambda-1}} f - \gamma_{B^\alpha_\lambda} f \right) \quad f : D \to \mathbb{R}, \tag{6}$$

## 3 Principle of the 1-D Opening Algorithm

We describe the main principles of our algorithm in this section. We begin with the description of the opening, the main principle of peak elimination, and its pseudocode. Later, we enrich this algorithm by feature of computing the size spectrum in a single image scan. We also present the arbitrary angle orientation on 2-D image support.

The proposed algorithm is based on usage of a queue, which is a FIFO (First In, First out) memory structure. In addition to the basic FIFO features *push()*, *pop()*, queue

provides *front()* and *back()* operations to access the oldest and most recent values, respectively. Each element stored in the queue is composed of two attributes $\{F, rp\}$: the pixel graytone value $F = f(rp)$ and its reading position $rp$ in input data stream. Both attributes can be accessed separately, i.e., $F_x = Q.back()[1]$ and $rp_x = Q.back()[2]$.

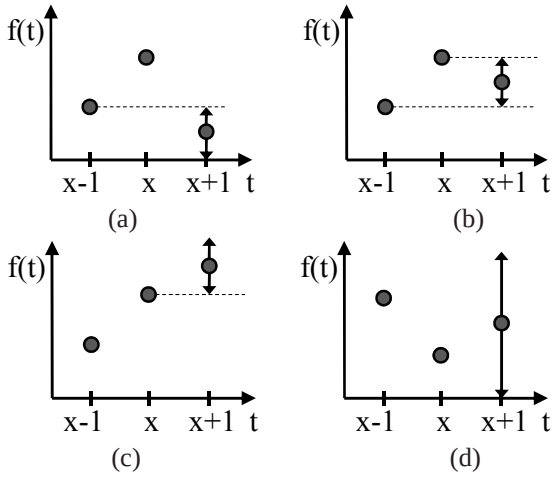



Figure 1: Four different pixel configuration for peak identification. Configuration (a) and (b) characterize a peak, whereas configuration (c) and (d) do not.

The behavior of this algorithm stems from the principle of a peak elimination. A point of an input signal $f(x)$ is a peak if both its very precedent $f(x-1)$ and subsequent $f(x+1)$ points are smaller (7). Our algorithm recognizes 4 possible configurations of these three points ((a) and (b) are peaks, (c) and (d) are not), see Fig. 1, each of which is treated in a different manner.

$$f(x) > f(x-1) \text{ and } f(x) > f(x+1) \tag{7}$$

The proposed algorithm, see Alg. 1 for the pseudocode listing, computes the opening $y = \gamma_B f$, $y, f \subset \mathbb{Z}^2 \rightarrow \mathbb{R}$, $B \subset \mathbb{Z}$ is a line segment of length $L$. It reads input pixel $F = f(rp)$ and outputs pixel $Y = y(rp - L)$ of the opened image at the time. The points $f(x-1)$, $f(x)$, and $f(x+1)$, which are needed for the peak elimination, are reachable in the queue as follows:

- $f(x-1) \rightarrow$ Q.back(2)[1] (second to the most recent)
- $f(x) \rightarrow$ Q.back(1)[1] (most recent)
- $f(x+1) \rightarrow F = f(rp)$ (current)

The peak elimination step proceeds in one main while loop (code line 1) that ensures the condition $f(x+1) < f(x)$ of (7). If two consecutive pixels are equal $f(x+1) = f(x)$, the first one is erased from the queue (line 2-4), and replaced by the second later (on line 12). As a consequence, a flat plateau (zone of constant value) is represented in the queue by the last pixel and its position.

Then the second condition $f(x-1) > f(x)$ of (7) is tested (Q.back(2)[1]< Q.back(1)[1] on line 6). If the result is false, the $f(x)$ is not a peak and the elimination loop is quit (configuration (d), line 14). Otherwise, the $f(x)$ is a peak and will

**Algorithm 1:** $Y \leftarrow$ 1D_OPEN ($F$, $rp$, $L$, Q)

**Input**: $F$ - input sample $f(rp)$; $rp$ - current reading position; $L$ - SE size; Q - Queue
**Result**: $Y$ - sample of $y(rp - L)$; $S$[] - Size spectrum
**Data**: Q - a Double-End Queue
Q.back(1)[] - accesses the last enqueued pair $\{F, rp\}$
Q.back(2)[] - accesses the second to the last enqueued pair $\{F, rp\}$

```
1  while F ≤ Q.back(1)[1] do
2      if F = Q.back(1)[1] then
3          Q.dequeue() ;                                   // Remove equal values
4          break ;
5      else
6          if Q.back(2)[1]<Q.back(1)[1] then
7              if F < Q.back(2)[1] then
8                  S[Q.back(1)[2] - Q.back(2)[2]] += Q.back(1)[1] - Q.back(2)[1] ;
                   // Accumulate discarded peak in size spectrum array S
9                  Q.back(2)[2] = Q.back(1)[2] ;                    // Config. (a)
10             else
11                 S[Q.back(1)[2] - Q.back(2)[2]] += Q.back(1)[1] - F ; // Accumulate peak
                   in S
12             Q.dequeue() ;                          // Discard peak, conf. (b), (a)
13         else
14             break ;                                          // Configuration (d)
15 Q.push({F, rp}) ;                                  // Enqueue current sample
16 if rp = Q.front()[2] + L then
17     Q.pop() ;                                      // Delete outdated value
18 if rp ≥ L then
19     return (Q.front()[1] ) ;                       // Return opening sample
```

be erased from the queue (line 12) and replaced by either $f(x-1)$ in configuration (a) (Q.back(2)[2] = Q.back()[2] on line 9), or by $f(x+1)$ in configuration (b) (needs only line 12). This is decided upon condition $f(x+1) < f(x-1)$ ($F <$ Q.back(2)[1] on line 3). Obviously, the while loop iterates until a non-peak configuration ((c) or (d)) is encountered.

When all peaks are erased, the current pixel value is unconditionally pushed into the queue along with current reading position (line 15). The oldest stored pixel is checked whether it has been stored in the queue for too long. This check is carried out by comparing the stored reading position plus SE size with current $rp$ (line 16). Outdated values are immediately deleted. The oldest stored value Q.front(1)[1] is the result of Alg. 1 as soon as $rp$ exceeds the SE size (line 18).

The algorithm presented so far computes the opening transformation using the principle of the peak elimination that discards the peak values. When we extend the algorithm by a feature for storing the eliminated peaks, we obtain the size spectrum $S$ with a minimal additional effort. As the algorithm eliminates a peak recursively, it literally slices the peak by each gray level of a peak. The slices of the same size are

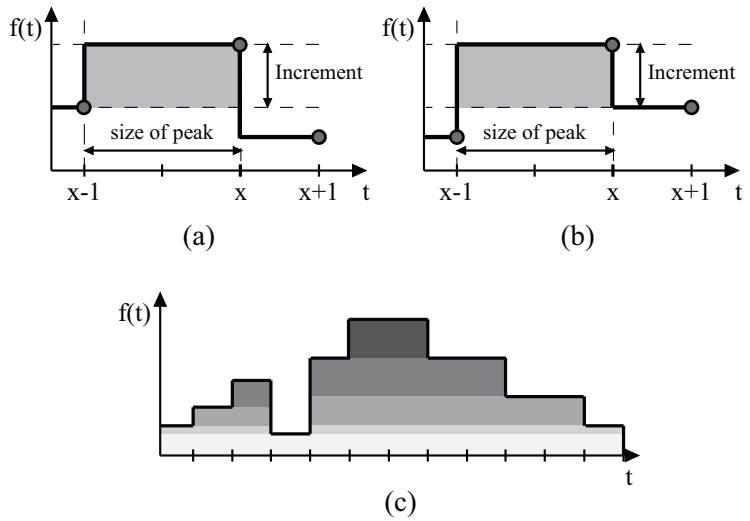



Figure 2: Size spectrum increment for (a) configuration a, and (b) configuration b. (c) The peak is sliced by each gray level it contains.

then accumulated in a single variable. Then the size spectrum $S$ is an array, such as $S[1..L-1] \rightarrow \mathbb{R}^+$ The size is determined by the distance between $f(x)$ and $f(x-1)$ that need not necessarily be 1 but varies from 1 to L-1 ($f(x \pm 1)$ designates precedent/subsequent value of $f(x)$). The height of a peak obviously depends on the mutual position of $f(x+1)$ and $f(x-1)$ according to conf. (a), (b), see Fig. 2. The increment of an appropriate size in the size spectrum is carried out on lines 8 and 11.

## 3.1 Arbitrary SE Orientation

The described algorithm can be used for a 2-D input image support as well. It only needs an image to be "mapped" into independent, 1 pixel thin discrete lines (called corridors) oriented in the same angle as the SE. See Fig. 3 for examples of such mapping. Each corridor is then computed according to 1D_OPEN in its own queue memory. The major advantage of mapping an image into corridors is that every pixel of input image is read just once, and the result pixel is written within a fixed delay. Furthermore, the input and output data are ordered in the horizontal scan pattern. It allows a stream, in-place, and, using a dedicated hardware, even parallel processing.

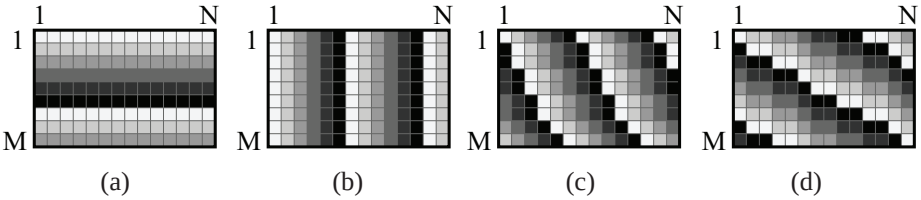



Figure 3: Image corridors (discrete lines) mapping for different SE orientations (a) a horizontal SE, (b) a vertical SE, and (c) and (d) inclined SEs.

The corridors are mapped on image by either of two cases in dependence on $\alpha$. For

$\alpha \geq 45°$ and $\alpha \leq 135°$, pixel [$j$,$i$] (i.e., at line $j$, column $i$) belongs to corridor $\mathrm{Q}_{ij} = (i - j \sin\alpha)\mathrm{modulo}(N + L \sin\alpha)$, where $M + L \sin\alpha$ defines the number of queues. The current reading position $rp$ within a corridor is equal to the line index $j$. On the other hand, for $\alpha < 45°$ and $\alpha > 135°$, pixel [$j$,$i$] (i.e., at line $j$, column $i$) belongs to corridor $\mathrm{Q}_{ij} = (j - i \sin\alpha)\mathrm{modulo}(N \tan\alpha + L \cos\alpha)$, where $(N + L) \tan\alpha + 1$ defines the number of queues, and the column index $i$ denotes the reading position $rp$.

Since each corridor has an independent queue, the image can be read in horizontal scan order regardless its orientation. The image is processed in an ordinary horizontal double-loop, such as *for* $j$ *in 1 to* $M + L \sin\alpha$ iterate *for* $i$ *in 1 to* $N + L \cos\alpha$ that calls $y(rp - L) \leftarrow \mathrm{1D\_OPEN}(f(rp), rp, L, \mathrm{Q}_{ij})$ of Alg. 1.

Let us observe different image configurations with respect to the SE orientation in Fig. 4. In the horizontal case, the scan order and the SE are parallel so the input is fetched in the right order as the algorithm needs. Therefore, a single queue is used. It reads a value at [$k$,$j$] and outputs in the same line at [$k$,$i$].

In the vertical or inclined case, the unlike scan and SE orientations require multiple queues. For instance, the inclined orientation in Fig. 4 reads a value at position [$l$,$j$] and returns a result at [$k$,$i$]. All pixels between these two points are stored in the queues implying the computation latency. Nevertheless, it is the minimal achievable latency considering unlike orientations of the SE and the scan order.

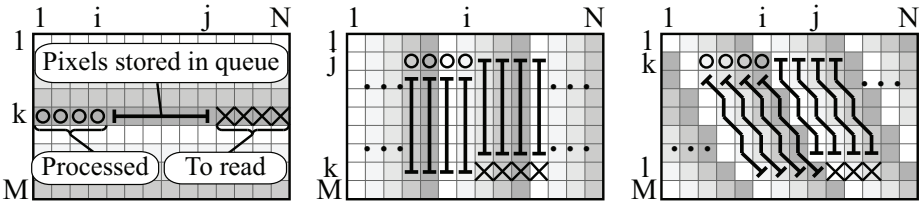



Figure 4: Image configuration for different SE orientations (a) a horizontal SE, (b) a vertical SE, and (c) an inclined SE. $\times$ denotes the pixels to be read in next iterations, $\circ$ denotes the previous output pixels.

The proposed algorithm has very limited memory consumption. The only memory elements are queues whose depth and count are inferred by the SE orientation and image width $N$. For example, let us consider an opening of 8-bit SVGA image, i.e., 800$\times$600 px by a SE of 41 px. The queues occupy only 656 bits for horizontal, and 525 kbits for vertical SE, respectively, compared to the mere size of the image 3.66 Mbits which is never stored.

# 4 Experimental Results

We present the execution time measures to illustrate the computational complexity in this section. We compare the proposed algorithm with a few other efficient algorithms, namely Soille *et al.* [5], Van Droogenbroeck [6], and Urbach and Wilkinson [7]. The benchmarks were performed on Intel Xeon E5620 @2.4GHz CPU, running Linux. The time reported in tables below refers to the user CPU time consumed by the respective algorithms. We use the mountain natural photo as a testing image, originally introduced at [6].

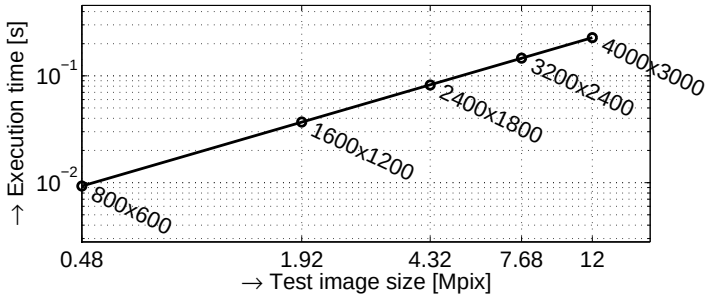



Figure 5: Execution time of opening/spectrum versus the image size. Structuring element is vertical 101 px.

At first, we evaluated the execution time benchmark with respect to the increasing image size, see Fig. 5. The results illustrates that the complexity of our algorithm is linear with the image size.

The second benchmark in Fig. 6 retains the same image size and changes the length of the horizontal SE to show that the execution time is independent of SE size (however the value of execution time is dependent on the image content). Compared with other algorithms, only Van Droogenbroeck's opening algorithm outperforms our solution. On the other hand, our algorithm computes the size spectrum in addition to the opening and is independent of the number of gray levels. For example, let us consider the size spectrum up to $\lambda_{\mathrm{MAX}} = 100$. Even if we omit the arithmetic operations of (4), the pure time for computation $\lambda_{\mathrm{MAX}}\ \gamma_\lambda$ will take 100×2.7 ms = 270 ms using the fastest Van Droogenbroeck's algorithm. Our algorithm computes the size spectrum in a single run, i.e., in 9.9 ms.

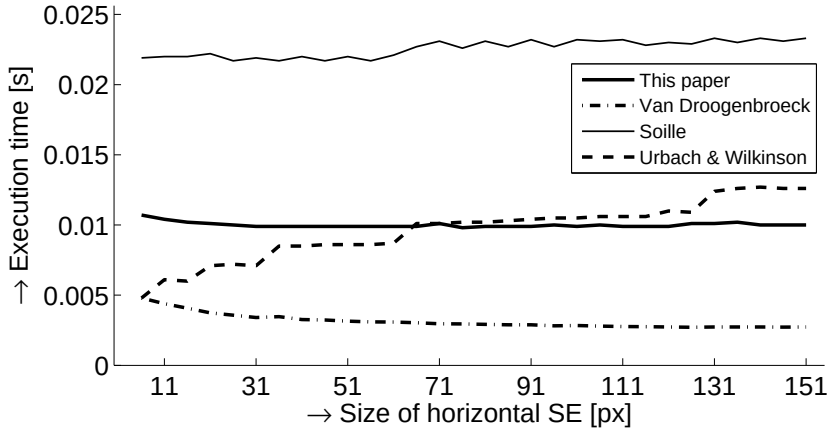



Figure 6: Execution time of opening versus the size of the horizontal structuring element. Natural photo 800 × 600 px is used.

The experiment in Fig. 7 reveals an influence of the SE rotation angle to the execution time. The proposed algorithm exhibits a small variation of execution time for the different octants. It is caused by the different spatial relation between corridors and

horizontal scan. The Urbach and Wilkinson algorithm results in much larger variation as its decomposition of the SE into line chords is not suitable for rotating line SEs.

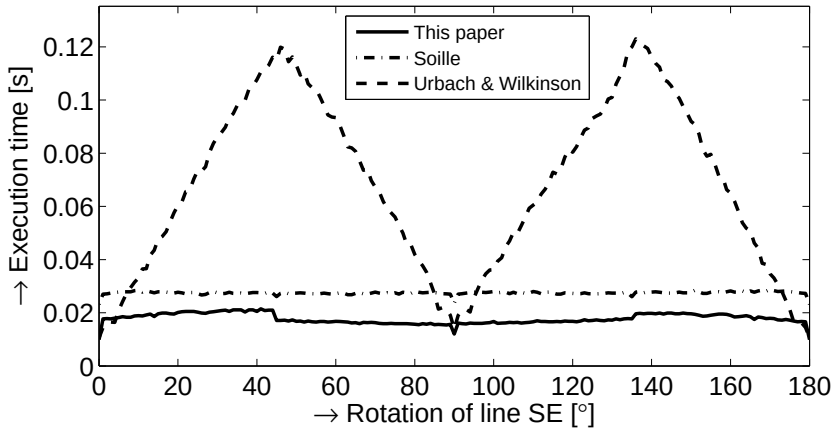


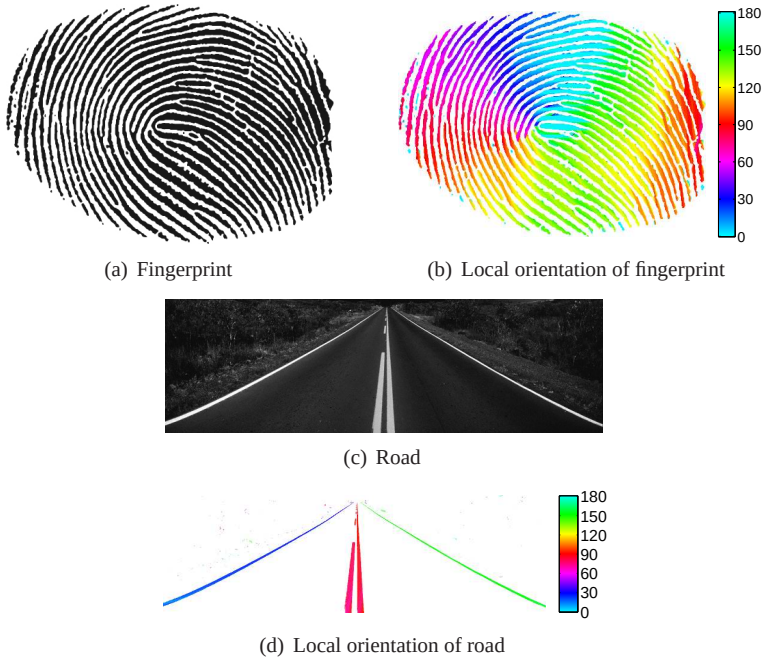

Figure 7: Execution time of opening versus the rotation angle $\alpha$. Natural photo 800 $\times$ 600 px is used.

# 5 Applications

This section deals with examples of suitable applications of arbitrary oriented opening and size spectrum. The first one is the local orientation information (orientation field) $\zeta_{35}$ of a fingerprint image in Fig. 8. The same operator can be applied to detect angle of the road lines in Fig. 8, too. Later, Fig. 9 displays the oriented size spectra $OS(\alpha, \lambda)$ for two different materials.

# 6 Conclusions

This paper presents a new algorithm for 1-D morphological opening and size spectrum extraction with large scale of suitable applications, such as anisotropic texture analysis, size distribution, local orientation, image restoration, etc.. The algorithm processes an image in a constant time regardless the SE size and almost regardless its rotation. The operator always uses strictly sequential access to data in horizontal scan pattern. Furthermore, it infers the minimal latency and very low memory requirements. The proposed algorithm achieves a high performance in the sequential C implementation and outperforms other arbitrary oriented line opening algorithms. Moreover, it computes the size spectrum in a single image scan introducing a speed-up in orders of magnitude compared to residue approach demanding multiple opening runs. The future extension of this work is to implement the proposed algorithm in the FPGA circuit taking an advantage of both spatial and temporal parallelism.




(a) Fingerprint (b) Local orientation of fingerprint

(c) Road

(d) Local orientation of road

Figure 8: Extraction of local orientation on (a) a fingerprint and (b) a road image.

# References

[1] Matheron, G.: Random sets and integral geometry [by] G. Matheron. Wiley New York, (1974)

[2] Maragos, P.: Pattern spectrum and multiscale shape representation. IEEE Trans. Pattern Anal. Mach. Intell. **11**(7) (1989) 701–716

[3] van Herk, M.: A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. Pattern Recogn. Lett. **13**(7) (1992) 517–521

[4] Gil, J., Kimmel, R.: Efficient dilation, erosion, opening, and closing algorithms. IEEE Trans. PAMI **24**(12) (2002) 1606–1617

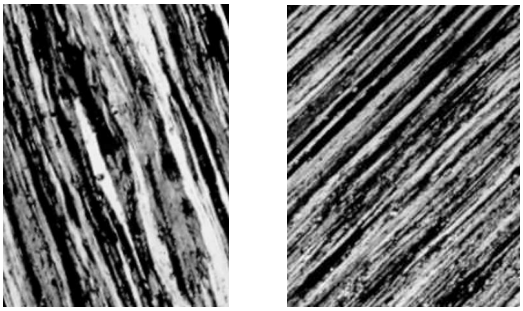
[5] Soille, P., Breen, E.J., Jones, R.: Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. IEEE Trans. Pattern Anal. Mach. Intell. **18**(5) (1996) 562–567

[6] Van Droogenbroeck, M., Buckley, M.J.: Morphological erosions and openings: Fast algorithms based on anchors. J. Math. Imaging Vis. **22**(2-3) (2005) 121–142

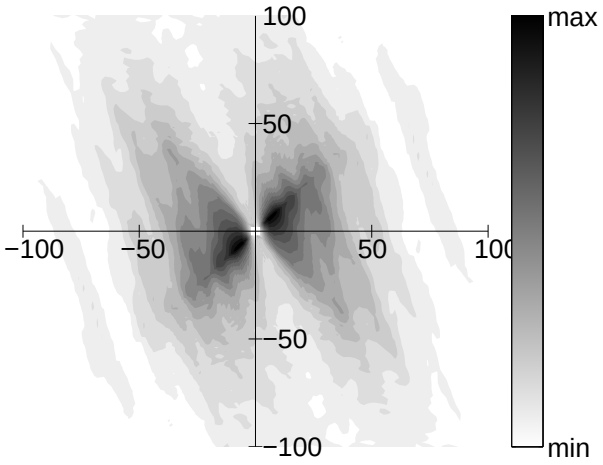
[7] Urbach, E., Wilkinson, M.: Efficient 2-d grayscale morphological transformations with arbitrary flat structuring elements. Image Processing, IEEE Transactions on **17**(1) (jan. 2008) 1 –8

[8] Vincent, L.: Granulometries and opening trees. Fundam. Inform. **41**(1-2) (2000) 57–90

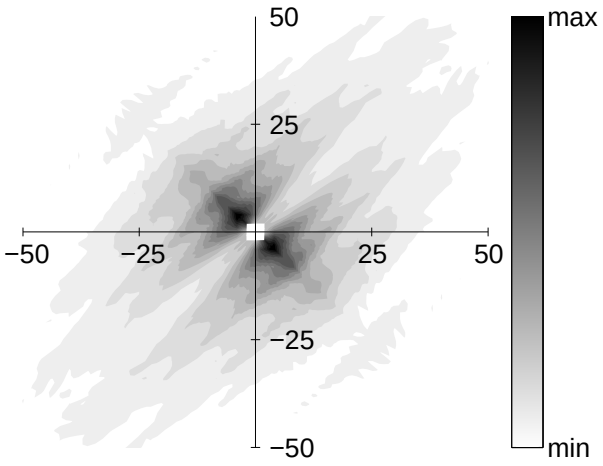
[9] Menotti-Gomes, D., Najman, L., de Albuquerque Araújo, A.: 1D Component tree in linear time and space and its application to gray-level image multithresholding. In: Proceedings of 8th ISMM. Volume 1., INPE (2007) 437–448



(a) Material A (b) Material B




(c) Size spectrum of material A




(d) Size spectrum of material B

Figure 9: Oriented size spectrum $OS(\alpha, \lambda)$. $\triangle\ \alpha = 1^{\circ}$, $\lambda_{\text{MAX}} = \{50, 100\}$.